Fall 9-1-1986

Results on NLC Grammars with One-Letter Terminal Alphabets ; CU-CS-348-86

Jochen Hoffmann
University of Colorado Boulder

Michael G. Main
University of Colorado Boulder

RESULTS ON NLC GRAMMARS WITH ONE-LETTER TERMINAL ALPHABETS*

Jochen Hoffmann and Michael G. Main

CU-CS-348-86 September 1986

*M. Main's research has been supported in part by National Science Foundation grant DCR-8402341.

RESULTS ON NLC GRAMMARS WITH ONE-LETTER TERMINAL ALPHABETS*

(September 1986)

Jochen Hoffmann and Michael G. Main
Department of Computer Science
University of Colorado
Boulder, CO 80309 USA
Phone: 303-492-7579

ABSTRACT

Node-label controlled graph grammars (NLC grammars) are a mechanism to generate sets of graphs (called graph languages). This paper examines the generating power of NLC grammars with a restricted connection relation and some closure properties of the class of NLC languages. Also, a modified language generating mechanism for NLC grammars is introduced, and the resulting class of graph languages is compared to the class of NLC languages. All results are based on grammars with one-letter terminal alphabets.

*M. Main's research has been supported in part by National Science Foundation grant DCR-8402341.

1. Introduction.

Recently, node-label controlled (NLC) graph grammars have been intensively studied as a method for generating sets of node-labelled graphs (called *graph languages*) [3,9]. The key feature of NLC-grammars is that both the application of a production and the embedding of a newly introduced subgraph are controlled by node-labels. These grammars were introduced and studied in a series of papers by Janssens and Rozenberg [3,4,5,6,7,8]. They examined the combinatorial structure of the languages generated by NLC grammars, and also researched several extensions and variations of the NLC model.

In a later paper, Ehrenfeucht, Main and Rozenberg [1] examined several restrictions on NLC grammars, including analogues of the Chomsky and Greibach normal forms for string grammars. None of the restrictions which they considered result in normal forms for NLC grammars. However, they left several questions unanswered concerning NLC grammars which have only a single terminal label. This paper resolves those questions, and also investigates additional properties of NLC grammars with a single terminal label. The additional properties involve closure results of the NLC languages, and an alternative generation mechanism which is motivated by parsing problems.

2. Preliminaries.

We will consider only finite undirected graphs without loops or multiple edges. The nodes of the graph are labelled with symbols from a finite alphabet. When all nodes of a graph α have the same label, we will sometimes write α as a pair $\alpha = (V, E)$, where V is the (finite) set of nodes in α and $E \subseteq (V \times V)$ is the set of edges in α. Since α is undirected and without loops, E must be a symmetric relation over V and cannot contain pairs of the form (v, v). We will refer to graphs satisfying

the above restrictions simply as graphs. If α is a graph, $|\alpha|$ denotes the number of nodes in α. For a finite alphabet Σ, $\mathbf{G}_\Sigma$ denotes the set of all graphs with node labels from Σ.

Definition. An NLC grammar is a system $\mathbf{G} = (\Sigma, \Delta, P, S, C)$, where

Σ is a finite nonempty set, called the set of *labels*,

Δ is a proper nonempty subset of Σ, called the set of *terminal labels*,

P is a finite set of *productions* :

each production has the form $X := \alpha$, where X is a nonterminal label ($X \in \Sigma - \Delta$) and α is a graph from $\mathbf{G}_\Sigma$,

$S \in \Sigma - \Delta$ is a special nonterminal label, called the *start symbol*,

$C \subseteq \Sigma \times \Sigma$ is a binary relation, called the *connection relation*.

Let $\mathbf{G} = (\Sigma, \Delta, P, S, C)$ be an NLC grammar. A production $(X := \alpha) \in P$ is applied as follows:

(1) Start with a graph μ and a specific occurrence of an X-labelled node in μ. This node is called the *mother node*. The set of nodes which are directly connected to the mother node is called the *neighborhood*.

(2) Delete the mother node from the graph μ.

(3) Add to μ a copy of the labelled graph α. This new occurrence of α is called the *daughter graph*.

(4) For each pair $(Y, Z) \in C$, connect every Y-labelled node in the daughter graph to every Z-labelled node in the neighborhood.

If η is the graph resulting from steps (1) - (4), we write $\mu \underset{G}{\Rightarrow} \eta$ to denote the relation "η is directly derived from μ in $\mathbf{G}$". A sequence of such transformations

$$\mathbf{D} = (\mu_0 \underset{G}{\Rightarrow} \mu_1 \underset{G}{\Rightarrow} \cdots \underset{G}{\Rightarrow} \mu_n)$$

is called a *derivation* of length n, and is denoted by $\mu_0 \overset{*}{\underset{G}{\Rightarrow}} \mu_n$. When the grammar $\mathbf{G}$ is understood, we also write simply $\mu \Rightarrow \eta$ or $\mu_0 \overset{*}{\Rightarrow} \mu_n$.

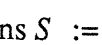
The language generated by the grammar $\mathbf{G}$ is the set of all graphs with terminal labels that can be derived from the graph with a single S-labelled node, i.e.,

$$\mathbf{L}(\mathbf{G}) = \{\mu \in \mathbf{G}_\Delta \mid S \overset{*}{\underset{G}{\Rightarrow}} \mu\}.$$

(We often use S to denote the one-node graph with label S.) A language L is called an *NLC language* if L is generated by some NLC grammar $\mathbf{G}$.

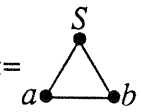
For any integer $k \geq 0$, an NLC grammar is called k-ary iff we have $|\alpha| \leq k$ for all productions $X := \alpha$. An ε-production is a production $X := \varepsilon$, where ε is the empty graph.

Examples.

Consider $\mathbf{G} = (\Sigma, \Delta, P, S, C)$, where $\Sigma = \{S, a\}$, $\Delta = \{a\}$, $C = \{(a, a)\}$ and P contains the two productions $S :=$  and $S :=$. The following is a derivation in $\mathbf{G}$:

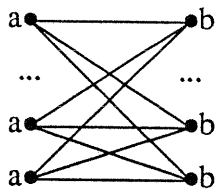
Note that in the last step the left node gets connected only to the middle node, since the right node was not in the neighborhood of the S-labelled node. The language generated by this grammar consists of all nonempty chains in which all nodes are labelled a.

As another example, let $\mathbf{G} = (\Sigma, \Delta, P, S, C)$, where $\Sigma = \{S, a, b\}$, $\Delta = \{a, b\}$, $C = \{(S,a), (S,b), (a,b), (b,a)\}$ and P contains these two productions:

$S :=$ 

$S :=$ 

In any derivation, the S-labelled node remains connected to all other nodes. Eventually, the second production must be applied, and the resulting graph is the final graph. At each step, the new a-labelled node must connect to all b-labelled nodes and vice versa. Therefore, the language generated by this grammar is $L = \mathbf{L}(\mathbf{G}) = \{B_{n,n} \mid n \geq 1\}$, where

$B_{n,n} =$ 

is a complete bipartite graph with n nodes on each side. All left nodes are labelled a and all right nodes are labelled b.

3. Restrictions on the Connection Relation.

One class of restrictions of NLC grammars examined in [1] concerns the connection relation of the grammar: An NLC grammar is called functional if its connection relation is a partial function. It is called inverse functional if the inverse of its connection relation is a partial function. And it is called symmetric if its connection relation is a symmetric relation.

It is shown in [1] that all of the above restrictions reduce the generating power of NLC grammars. However, the proofs assume that the terminal alphabet has a size ≥ 2. We show that the

results from [1] still hold for a terminal alphabet of size 1.

The proofs in this section assume NLC grammars without ε-productions. We therefore need as a preliminary result Theorem 2.1 from [1]:

> Let $k \geq 0$ be an integer and $\mathbf{G} = (\Sigma, \Delta, P, S, C)$ be a k-ary grammar. Then there exists a k-ary grammar $\mathbf{G}' = (\Sigma, \Delta, P', S, C)$ such that P' contains no ε-productions and $\mathbf{L}(\mathbf{G}') = \mathbf{L}(\mathbf{G}) - \{\varepsilon\}$. That is, $\mathbf{L}(\mathbf{G}')$ is identical to $\mathbf{L}(\mathbf{G})$, except possibly for the empty graph.

The transformation of an NLC grammar into an equivalent grammar without ε-productions can be done effectively: it is very similar to the elimination of ε-productions from a context-free string grammar [2]. Note that the elimination of ε-productions changes only the set of productions P: no other components of the grammar are affected. In particular, this implies that ε-productions can be eliminated without changing properties of the connection relation C (like functionality, inverse functionality and symmetry). We sometimes call an NLC grammar without ε-productions an *ε-free* grammar.

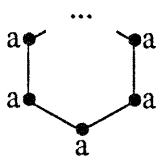
Derivations in a grammar without ε-productions have the important property that the number of nodes is non-decreasing from one derivation step to the next. For a derivation $\mathbf{D} = (\overset{S}{\bullet} \Rightarrow \cdots \Rightarrow \mu)$ with $n = |\mu| > 1$, it follows that there is a last graph γ with fewer than n nodes. We can therefore write

$$\mathbf{D} = (\overset{S}{\bullet} \Rightarrow \cdots \Rightarrow \gamma \Rightarrow \delta \Rightarrow \cdots \Rightarrow \mu),$$

where all graphs up to γ have fewer than n nodes, and all graphs after γ have exactly n nodes. For such a derivation, we call γ the *critical graph*, and we call the derivation step $\gamma \Rightarrow \delta$ the *critical step*.

Theorem 1. There is an NLC language L over a one-letter terminal alphabet, which is not generated by any functional NLC grammar.

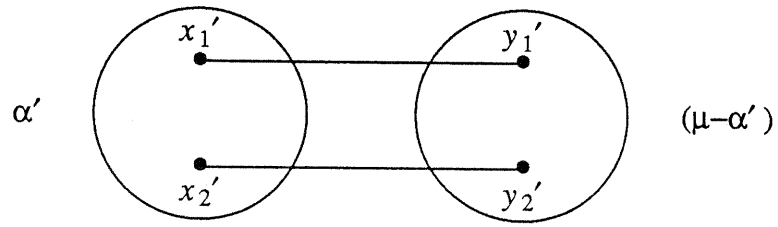
Proof: Let $L = \mathbf{G}_{\{a\}} - \{\varepsilon\}$, the language of all nonempty graphs with nodes labelled a. It is easy to verify that L is an NLC language. Suppose, there is a k-ary ($k \geq 2$) functional NLC grammar $\mathbf{G} = (\Sigma, \{a\}, P, S, C)$ that generates L. We may assume that P contains no ε-productions. Also note that $(a, a) \in C$, since otherwise no graph generated by $\mathbf{G}$ could contain a connected component with more than k nodes. L contains all circles of the form

$C_n =$  with $n \geq 3$ nodes.

Choose $n > k+2$ and let $\mu = \mathbf{C}_n$. We will now show that the assumption of $\mathbf{G}$ deriving μ leads to a contradiction, which proves the theorem.

Let $\mathbf{D} = (\overset{S}{\bullet} \Rightarrow \cdots \Rightarrow \gamma \Rightarrow \delta \Rightarrow \cdots \Rightarrow \mu)$ be a derivation of μ in $\mathbf{G}$, where γ denotes the critical graph, i.e., $|\gamma| < n$, $|\delta| = n = |\mu|$. Let $X := \alpha$ be the production applied in the step $\gamma \Rightarrow \delta$. Since this step increases the number of nodes, $|\alpha| \geq 2$. After this step, only node-relabelling productions are applied.

Let α' be the subgraph of the final graph μ derived from α. Since μ is a circle with $n > |\alpha'|$ nodes, there must be two nodes x_1' in α' and y_1' in $\mu - \alpha'$ which are directly connected. Since $|\alpha'| = |\alpha| \geq 2$ and $|\mu - \alpha'| = |\mu| - |\alpha| \geq n - k \geq k + 2 - k = 2$, we can choose two more nodes x_2' in α' and y_2' in $(\mu - \alpha')$, such that $x_2' \neq x_1'$, $y_2' \neq y_1'$, and x_2', y_2' are directly connected:



Let x_1, x_2, y_1, y_2 be the nodes in δ which are the ancestor nodes of x_1', x_2', y_1', y_2' respectively. Then, x_1, y_1 and x_2, y_2 must be directly connected in δ. Now we show that y_1 and y_2 are both labelled a in δ: Suppose, y_1 is labelled $Y \neq a$.

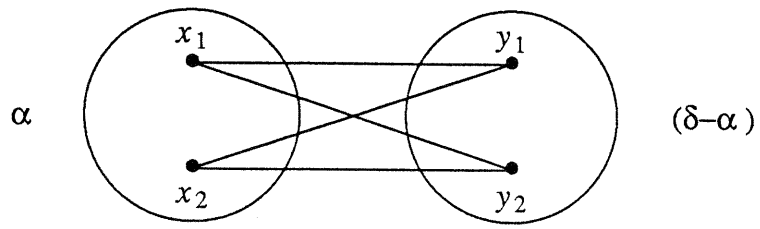
case 1. x_1 is labelled a.

Since x_1 is connected to y_1, we have $(a, Y) \in C$. Since $(a, a) \in C$, this contradicts the functionality of G.

case 2. x_1 is labelled $X \neq a$.

Then x_1 and y_1 both have nonterminal labels. Either x_1 or y_1 must derive an a-labelled node first (in the derivation **D**). At this step, the edge connecting x_1 and y_1 must be broken (since **G** is functional and $(a, a) \in C$, thus $(a, Z) \notin C$ for all $Z \neq a$). This is a contradiction to x_1', y_1' being connected in μ.

Therefore, y_1 must be labelled a in δ, and similarly y_2 must be labelled a in δ. Since **G** is NLC, there must also be edges connecting x_1, y_2 and x_2, y_1 in δ.



Since any rewriting of x_1 (or x_2) treats the connections to y_1 and y_2 identically, all four edges must be present in the final graph μ. But n was chosen > 4, so μ cannot have a cycle of length 4.

□

Theorem 2. There is an NLC language L over a one-letter terminal alphabet, which is not generated by any inverse functional NLC grammar.

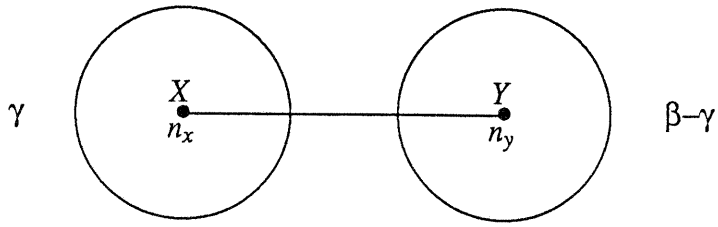
Proof: A graph $g = (V, E)$ is called complete iff

$$E = (V \times V) - \{(v,v) | v \in V\},$$

i.e., if it has all possible edges. Let L be the language of all complete graphs with nodes labelled a. It is easy to verify that L is an NLC language. Assume, $\mathbf{L}(\mathbf{G}) = L$ for a k-ary ε-free inverse functional NLC grammar **G**.

Let $\mu \in L$ be a sufficiently large ($|\mu| > k^2$) complete graph. Let $\mathbf{D} = (S \Rightarrow \cdots \Rightarrow \alpha \Rightarrow \beta \Rightarrow \cdots \Rightarrow \mu)$ be a derivation of μ in **G**. Let $\alpha \Rightarrow \beta$ be the last step in **D** where a production is applied which contains a non-terminal in the daughter graph. Let $Z := \gamma$ be the production applied in this step. Let X be a nonterminal label of a node n_x in γ. The graph $(\beta - \gamma)$ cannot be empty, otherwise $|\beta| = |\gamma| \leq k$, which would imply $|\mu| \leq k^2$. Therefore, we can choose a node $n_y \in (\beta - \gamma)$. Let Y be the label of n_y. Since the final graph is complete, n_x must

connect to n_y, and we have $(X, Y) \in C$.



Let $\mathbf{D}'$ be a derivation in **G** obtained from **D** as follows:

If Y is a nonterminal label, and n_y is rewritten in the tail ($\beta \Rightarrow \cdots \Rightarrow \mu$) of **D** before n_x is rewritten, then swap the two derivation steps which rewrite n_x and n_y, otherwise let $\mathbf{D}'=\mathbf{D}$.

In the resulting derivation $\mathbf{D}'$, n_y is still labelled Y when n_x is rewritten. By the choice of the step ($\alpha \Rightarrow \beta$) (and since **G** is ε-free), the rewriting of n_x introduces one or more a-labelled nodes. Since **G** generates only complete graphs, these nodes must connect to n_y, hence $(a, Y) \in C$.

We have $(a, Y) \in C$ and $(X, Y) \in C$. But since $X \neq a$, **G** cannot be inverse functional. The contradiction shows that L cannot be generated by an inverse functional NLC grammar.

□

To show that symmetry of the connection relation also reduces the generating power of NLC grammars for a one-letter terminal alphabet, we need a few definitions and two technical lemmas:

Definition. Let $g = (V, E)$ be a graph and X a subset of the nodes of g.

a. $\mathbf{N}(X) = \{v \in V \mid$ there exists an $x \in X : (v, x) \in E\}$

= all nodes that are directly connected to some node in X

b. $Ne(X) = \mathbf{N}(X) - X$

c. X is an *extreme set* in g iff for all nodes $x \in X$ either

i) x is directly connected to every node in $Ne(X)$,or

ii) x is not directly connected to any node in $Ne(X)$.

d. $x \in X$ is an *inner node of* X iff x is not directly connected to any node in $Ne(X)$.

Lemma 1. Let L be an NLC language over a one-letter terminal alphabet and **G** a k-ary, ε-free NLC grammar that generates L. Then, for any $g \in L$ (with $|g| > k$), there exists a $g' \in L$ such that

a) $|g'| = |g|$

b) g' has an extreme set X with $2 \leq |X| \leq k$

Proof: Let $g \in L$ (with $|g| > k$), and let

$\mathbf{D} = (S \Rightarrow \cdots \Rightarrow \beta \Rightarrow \gamma \Rightarrow \cdots \Rightarrow g)$ be a derivation of g in **G**,

$(\beta \Rightarrow \gamma)$ the critical step in **D**,

$Z := \alpha$ the production applied in the critical step,

z the Z-labelled node in β that is rewritten in the critical step,

and $Ne(z)$ the neighborhood of z in β.

Rearrange the steps of the derivation **D** as follows:

(1) copy all steps up to (but not including) the critical step;

(2) rewrite the nodes in $Ne(z)$ until they are all labelled by the one terminal label (applying steps from the latter part of the derivation);

(3) apply the critical step;

(4) apply the remaining steps from **D**.

The graph g' resulting from the modified derivation is in L and has the same number of nodes as g. Let X be the set of nodes in g' that was introduced in the critical step.

We have $2 \leq |X| \leq k$. Moreover, X is extreme: Let x be a node in X. When x is introduced in the critical step, all nodes in $Ne(X) = Ne(z)$ are labelled a (the terminal label). Since **G** is NLC, x is connected to either none or all the nodes in $Ne(X)$. If, in the remainder of the modified derivation, x is disconnected from one of the nodes in $Ne(X)$, then it must disconnect from all of them.

□

Lemma 2. Let L be an infinite NLC language of connected graphs over a one-letter alphabet. Let $\mathbf{G} = (\Sigma, \{a\}, P, S, C)$ be a symmetric k-ary, ε-free NLC grammar with $\mathbf{L}(\mathbf{G}) = L$.
Then, for all $g \in L$ with $|g| > k$, there is a graph $g' \in L$ with the following properties:

(1) $|g'| = |g|$

(2) g' contains an extreme set X with $2 \leq |X| \leq k$, such that no two inner nodes of X are connected.

Proof: Let $g \in L$ with $|g| > k$, and let $Z := \alpha$ be the production applied in the critical step in a derivation of g in **G** ($2 \leq |\alpha| \leq k$). Rearrange the derivation as follows:

(1) apply all the steps up to (but not including) the critical step;

(2) let z be the node which will be rewritten in the critical step: apply relabelling productions from the latter part of the derivation to the nodes in the neighborhood of z, until they all have terminal labels;

(3) apply the critical step;

(4) select a node that still has a nonterminal label (if there is one) and apply the remaining steps for this node in sequence;

(5) repeat (4) in turn for every remaining node that still has a nonterminal label.

Let g' be the resulting graph, and let X be the set of nodes in g' introduced in the critical step. We have $|g'|=|g|$, and X is an extreme set in g' with $2 \leq |X| \leq k$ (see the proof of Lemma 1). It remains to show that no two inner nodes in X are connected by an edge.

Let u,v be two inner nodes in X and let $N = Ne(X)$. Since g' is connected and $|g'| > k$, we have $N \neq \emptyset$. Since u and v are not directly connected to any node in N, and since all the nodes in N are labelled a when the critical step is applied, there are labels $U,V \in \Sigma$ with the following properties:

(1) $(U,a) \notin C$ and $(V,a) \notin C$;

(2) in some intermediate graph after the critical step, u is labelled U;

(3) in some intermediate graph after the critical step, v is labelled V.

Since $|g'| > k$ and g' is connected, we have $(a,a) \in C$, and therefore $U \neq a$ and $V \neq a$. Since the production steps after the critical step are ''not mixed'', one of the nodes u,v must be labelled a before any production is applied to the other node. Let u be the node that reaches the terminal label first.

case 1. When the final step is applied to u, v is labelled V. Node u must disconnect from v at this step, since $(V,a) \notin C$ and C is symmetric, therefore $(a,V) \notin C$.

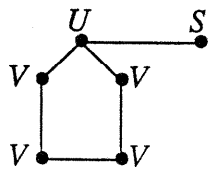
case 2. When the final step is applied to u, v is not labelled V. Since no productions have been applied to v yet, v must reach label V when u is already labelled a. At this step, v disconnects from u.

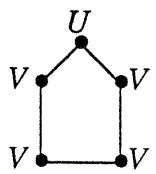
□

Theorem 3. There is an NLC language L over a one-letter alphabet which is not generated by any symmetric grammar.

Proof: Let $\mathbf{G} = (\Sigma, \{a\}, P, S, C)$, where $\Sigma = \{S, U, V, a\}$,

P contains the following productions:

(1) $S :=$ 

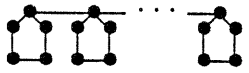
(2) $S :=$ 

(3) $U :=$ 

(4) $V :=$ 

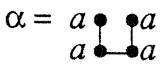
$C = \{(a,a), (a,U), (a,V), (U,a), (U,U)\}$.

$L = \mathbf{L}(\mathbf{G})$ consists of all graphs of the form

$g =$  (all nodes labelled a)

Suppose, there is a symmetric k-ary NLC grammar $\mathbf{G}'$ with $\mathbf{L}(\mathbf{G}') = L$. We may assume that $\mathbf{G}'$ is ε-free. Let $g \in L$ with $|g| > k+3$. Since L contains only one graph with $|g|$ nodes, g must contain an extreme set X with $2 \leq |X| \leq k$, such that no two inner nodes of X are directly connected (by Lemma 2).

However, an analysis by cases shows that any extreme set in g with ≥ 2 nodes contains at least one subgraph α of the form

$\alpha =$ 

Therefore, any extreme set in g with ≥ 2 nodes contains at least one pair of inner nodes which are directly connected. The contradiction shows that L cannot be generated by a symmetric grammar.

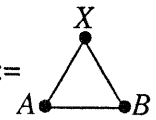
□

4. Closure Properties of NLC.

In this section, we examine closure properties of the class of NLC graph languages: we consider the set-theoretic operations intersection and complement and the graph-theoretic operations *dual* and *edge complement.*

Theorem 4. The class NLC is not closed under intersection.

Proof: Let $\mathbf{G}_1 = (\Sigma, \{a\}, P, S, C_1)$, $\mathbf{G}_2 = (\Sigma, \{a\}, P, S, C_2)$, where $\Sigma = \{S, A, B, X, a\}$, P contains the following productions:

(1) $S :=$ 

(2) $X :=$ 

(3) $A :=$ 

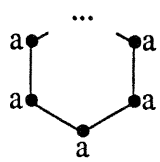
(4) $X :=$ 

(5) $B :=$ 

$C_1 = \{(A, A), (X, B), (a, \Sigma), (\Sigma, a)\}$,

$C_2 = \{(A, A), (X, B), (a, \Sigma)\}$.

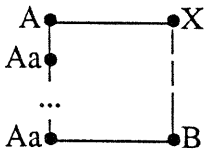
First, we show that $L_1 = \mathbf{L}(\mathbf{G}_1) = \mathbf{C} \cup R_1$, where $\mathbf{C} = \{C_3, C_4, \cdots\}$,

($C_n =$ 

, a circle with n nodes)

and all graphs in R_1 contain a cycle of length 3:

Every derivation in $\mathbf{G}_1$ must start with production (1), followed by $n \geq 0$ applications of production (2) and $\leq n$ applications of production (3) to nodes not adjacent to the X-labelled node. The resulting graph is

$\alpha_n =$ 

where Aa denotes either label A or label a, and the graph has $n+3$ nodes. Clearly, $\alpha_n \overset{*}{\underset{G_1}{\Rightarrow}} C_{n+3}$. In order to derive from α_n a graph which is not in **C**, one of the neighbors of the X-labelled node must be rewritten, followed by one or more applications of production (2). Since every label connects to a, this will introduce a cycle of length 3 as a subgraph.

Next, we show that $L_2 = \mathbf{L}(\mathbf{G}_2) = \mathbf{C} \cup R_2$, where R_2 contains only acyclic graphs: As in $\mathbf{G}_1$, $S \overset{*}{\underset{G_2}{\Rightarrow}} \alpha_n$, and therefore $\mathbf{C} \subseteq L_2$. To derive from α_n a graph which is not in **C**, one of the neighbors of the X-labelled node must be rewritten, followed by one or more applications of production (2). Since the labels A and X do not connect to a in $\mathbf{G}_2$, this will break the cycle. Productions (2)-(5) cannot introduce a cycle, so the graph remains acyclic.

$R_1 \cap R_2 = \varnothing$, so $L_1 \cap L_2 = \mathbf{C}$. It is shown in [1] that **C** is not an NLC language, so NLC is not closed under intersection.

□

Corollary . The class NLC is not closed under set complement.

Proof: This follows immediately from theorem 4 and the fact that NLC is closed under set union.

□

Definition . For a graph $g \in \mathbf{G}_{\{a\}}$, let *dual* (g) be the graph resulting from the following construction:

(1) Represent every edge in g by a node in *dual* (g), labelled by a.

(2) Connect any two nodes in *dual* (g) by an edge iff the corresponding edges in g are adjacent.

For a graph language $L \subseteq \mathbf{G}_{\{a\}}$, let *dual* $(L) = \{dual(g) \mid g \in L\}$.

Theorem 5. The class NLC is not closed under the operation *dual*.

Proof: Let L be the language of all complete graphs with nodes labelled by a (as in Theorem 2). It is straightforward to show that L is NLC, but *dual* (L) is not NLC, since the size of the graphs in *dual* (L) does not grow in steps bounded by a constant, as required by Corollary 2 from [3].

□

Definition. Let $g \in \mathbf{G}_{\{a\}}$ be a graph. The *edge complement* $\overline{g}$ of g is the graph in $\mathbf{G}_{\{a\}}$, which has the same set of nodes as g and has an edge connecting any two distinct nodes u and v if and only if u and v are not connected by an edge in g. For a graph language L, the *edge complement* $\overline{L}$ of L is the set $\{\overline{g} \mid g \in L\}$.

Theorem 6. The class NLC is not closed under the operation *edge complement*.

Proof: The grammar

$$\mathbf{G} = (\{S, a\}, \{a\}, \{S := \overset{a}{\bullet}\!\!-\!\!\bullet\ S\ ,\ S := \overset{a}{\bullet}\}, S, \{(a,a)\})$$

generates the language L of all non-empty chains of a-labelled nodes. Suppose, there is a k-ary

NLC grammar ($k \geq 2$) that generates $\bar{L}$. Let $n \geq k+3$ and $\bar{g} \in \bar{L}$ be a graph with n nodes. Let $V = \{v_1, v_2, \ldots, v_n\}$ be the set of nodes of $\bar{g}$. The numbering of the nodes is obtained by considering g as a left-to-right oriented chain of nodes. Finally, let X be an extreme set of nodes in $\bar{g}$ with $2 \leq |X| \leq k$. (Such a subset exists by Lemma 1.)

Note that $|V - X| > 2$, which implies that every node in X is connected (in $\bar{g}$) to at least one node outside of X. (This is because every node in $\bar{g}$ is disconnected from at most two other nodes.) Thus (since X is extreme), every node in X is connected (in $\bar{g}$) to every node in $Ne(X)$. Now, consider two nodes v_i and v_j in X, with $i < j$. If $i \neq 1$, then the node v_{i-1} exists and is not connected (in $\bar{g}$) to v_i, which implies $v_{i-1} \notin Ne(X)$. But, v_j is connected (in $\bar{g}$) to v_{i-1}, which implies that $v_{i-1} \in X$. Continuing this argument shows that $v_1 \cdots v_i$ are all in X. Similarly, $v_j \cdots v_n$ and $v_{i+1} \cdots v_{j-2}$ are also all in X. But now, $|X| \geq n-1 > k$, which is a contradiction. By this contradiction, we conclude that $\bar{L}$ is not NLC.

□

5. Neighborhood-Preserving Derivations.

In this section we consider a language generating mechanism slightly different from NLC:

Definition.

(1) A derivation step is called *neighborhood preserving (NP)* iff the neighborhood of the daughter graph (not including the nodes of the daughter graph itself) is equal to the neighborhood of the mother node.

(2) A derivation is called *neighborhood preserving* if each of its derivation steps is neighborhood preserving.

(3) For an NLC grammar $\mathbf{G}$, the *neighborhood preserving language* of $\mathbf{G}$, $\mathbf{L}_{NP}(\mathbf{G})$, is the set of all graphs g, such that a neighborhood preserving derivation of g exists in $\mathbf{G}$. Note that in general $\mathbf{L}_{NP}(\mathbf{G}) \subseteq \mathbf{L}(\mathbf{G})$.

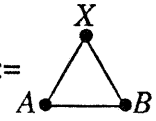
(4) The class *NPNLC* is the class of all graph languages L, such that, for some NLC grammar $\mathbf{G}$, $L = L_{NP}(\mathbf{G})$.

Given the graph resulting from a neighborhood preserving derivation step, the daughter graph and the production applied, the mother graph is uniquely determined. Since arbitrary NLC derivations don't have this property, languages in NPNLC seem to be easier to parse than NLC languages. We therefore examine the relationship between NLC and NPNLC.

Theorem 7. NPNLC is not a subclass of NLC.

Proof: Let $\mathbf{G} = (\Sigma, \{a\}, P, S, C)$, where $\Sigma = \{S, A, B, X, a\}$,

P contains the following productions:

(1) $S :=$ 

(2) $X :=$ 

(3) $A :=$ 

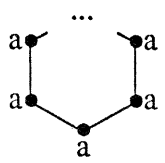
(4) $X :=$ 

(5) $B :=$ 

,

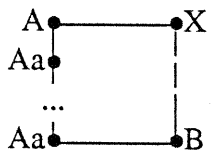
$C = \{(A,A),(X,B),(a,\Sigma)\}$.

We show that $\mathbf{L}_{NP}(\mathbf{G}) = \mathbf{C} = \{C_3, C_4, \ldots\}$, where

$C_n =$ 

is a circle with n nodes.

Every derivation in $\mathbf{G}$ must start with production (1), followed by $n \geq 0$ applications of production (2) and $\leq n$ applications of production (3) to nodes not adjacent to the X-labelled node. The resulting graph is

$\alpha_n =$ 

where Aa denotes either label A or label a, and the graph has $n+3$ nodes. When production (3) is applied to the A-labelled node which is adjacent to the X-labelled node, or when production (5) is applied, the X-labelled node gets an a-labelled neighbor. This implies that production (2) can not be applied again, since this step would not be neighborhood-preserving. Therefore, the size of the graph can not change any more, and only the relabelling productions (3,4,5) can be applied. Since a connects to any label, no connections are broken, and the final graph must be in $\mathbf{C}$. Conversely, every graph in $\mathbf{C}$ is in L. It is shown in [1] that $\mathbf{C}$ is not an NLC language, so $\mathbf{C} \in NPNLC - NLC$.

□

Theorem 8. NLC is not a subclass of NPNLC.

Proof: We first observe that the class NPNLC is closed under intersection with discrete graphs. A graph $g = (V, E)$ is discrete iff $E = \varnothing$. Let $\mathbf{D} \subseteq \mathbf{G}_{\{a\}}$ denote the set of all discrete graphs with nodes labelled a.

Consider a neighborhood preserving derivation: if, at any step, an edge is introduced, it cannot be broken and must be present in the final graph. Therefore, a neighborhood preserving derivation derives a discrete graph if and only if the daughter graph at each step is discrete.

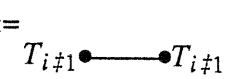
Let $L \in NPNLC$ and let $\mathbf{G} = (\Sigma, \{a\}, P, S, C)$ be an NLC grammar with $\mathbf{L}_{NP}(\mathbf{G}) = L$. By removing from P all those productions whose daughter graphs contain edges, we obtain a grammar $\mathbf{G}'$ with $\mathbf{L}_{NP}(\mathbf{G}') = L \cap \mathbf{D}$, i.e., $L \cap \mathbf{D} \in NPNLC$.

We can now prove the theorem by presenting an NLC language L, such that $L \cap \mathbf{D} \notin NPNLC$. In the description of the grammar, we use the notation $i \ddagger j$ to denote addition modulo 3. Let $L = \mathbf{L}(\mathbf{G})$ and $\mathbf{G} = (\Sigma, \{a\}, P, S, C)$ where

$\Sigma = \{a, S, R_0, R_1, R_2, T_0, T_1, T_2, L_0, L_1, L_2\}$,

P contains the following 13 productions:

(1) $S := \overset{T_0}{\bullet}$

(2) $T_i :=$  $(i = 0,1,2)$

(3) $T_i := \overset{L_i}{\bullet}$ $(i = 0,1,2)$

(4) $L_i := \overset{a}{\bullet}$ $(i = 0,1,2)$

(5) $R_i := \overset{a}{\bullet}$ $(i = 0,1,2)$, and

$C = \Sigma \times \Sigma - \{(R_i, R_i), (R_i, T_{i\ddagger 1}), (T_{i\ddagger 1}, R_i), (L_i, L_i) \mid 0 \le i \le 2\}$.

We are interested only in the discrete graphs in L. The derivation of a discrete graph in **G** proceeds in a way resembling the construction of a complete binary tree: At the lowest "level", we have some number of nodes labelled T_i (for some $i \in \{0,1,2\}$), which form a complete subgraph of the current graph. The levels above are filled with isolated nodes. Initially (i.e., after applying production (1) and (2) with $i = 0$), we have one isolated node labelled R_0 and two nodes labelled T_1, which form a complete subgraph of size 2. In order to derive a discrete graph, all edges on the lowest level must be broken. This can be done by rewriting all the T_i-labelled nodes in turn using production (3). Since $(L_i, L_i) \notin C$, all connections are broken, and we have a discrete graph. At this point, only the relabelling productions of type (4) and (5) can be applied.

Instead of using productions of type (3), we can choose to expand the "tree" by applying productions of type (2) to the T_i-labelled nodes on the lowest level. This starts a new level of nodes labelled $T_{i\ddagger 1}$ below the current level. Note that each such derivation step introduces an R_i-labelled node, which remains connected to all T_i-labelled nodes on the current level. These con-

nections can be broken only by applying production (2) to *all* T_i-labelled nodes on the current level. This property of the grammar forces the whole current level to be rewritten using production (2), which roughly doubles the number of nodes in the graph.

Moreover, in the derivation of a discrete graph, the ''tree'' can never become unbalanced: suppose that, at some stage of the derivation, the current level has been partially expanded, i.e., it still contains some T_i-labelled nodes, and the level below contains some nodes labels $T_{i\neq 1}$. If we would now apply production (2) (''prematurely'') to one of the $T_{i\neq 1}$-labelled nodes on the lowest level, we would introduce a connection of the form $\overset{T_i}{\bullet}\!\!-\!\!\!-\!\!\overset{R_{i\neq 1}}{\bullet}$, which can never be broken. Therefore, the graph resulting from such a derivation cannot be discrete.

Since any level in a discrete graph derived in **G** contains twice as many nodes as the previous level, we have

$$L \cap \mathbf{D} = \{g \in \mathbf{G}_{\{a\}} \mid g \text{ is discrete and } |g| = 2^n - 1 \text{ for some } n \geq 2\}.$$

The set $\{|g| : g \in L \cap \mathbf{D}\}$ has arbitrary large gaps, so corollary 2 from [3] can be used to show that $L \cap \mathbf{D}$ is not an NLC language (see proof of theorem 5). Moreover, $L \cap \mathbf{D}$ cannot be in NPNLC, since corollary 2 from [3] also applies to NPNLC (its proof is based on a pumping argument which is equally applicable under the NP restriction).

If L were in NPNLC, then, by the closure property shown above, $L \cap \mathbf{D}$ would have to be in NPNLC. Therefore, $L \in NLC - NPNLC$.

□

5. DISCUSSION.

We have shown that even for a one-letter terminal alphabet the functional, inverse functional and symmetric restriction on the connection relation reduces the generating power of NLC grammars. A primary technique in these proofs is the identification of the first point in a derivation where the number of nodes no longer grows (the "critical point"). By rearranging derivation steps which are nearby the critical point, graphs with special properties can be generated by the restricted NLC grammars. However, such graphs are not always generated by arbitrary NLC grammars, hence the restrictions are a reduction in power. It is still an open problem to find normal forms for NLC grammars, i.e. restrictions on NLC grammars that do not reduce their generating power.

Other questions of interest are: How can we characterize the class of symmetric (functional, inverse functional) NLC languages? What graph-theoretic operations preserve the NLC property? NPNLC is a class of graph languages resulting from a restriction on NLC derivations. We found that not all NLC languages can be generated this way, but we gain some non-NLC languages. Can efficient parsing algorithms be developed for NPNLC languages? Is the class NPNLC broad enough to be interesting for applications?

References

(1) A. Ehrenfeucht, M.G. Main and G. Rozenberg, Restrictions on NLC graph grammars, *TCS* 31 (1984), 211-223.

(2) M. Harrison, *Introduction to Formal Language Theory*, Addison-Wesley, Reading, MA., (1978).

(3) D. Janssens and G. Rozenberg. On the structure of node-label controlled graph languages, *Information Sciences* 20 (1980), 191-216.

(4) D. Janssens and G. Rozenberg. Restrictions, extensions and variations of NLC grammars, *Information Sciences* 20 (1980), 217-244.

(5) D. Janssens and G. Rozenberg. Decision problems for node-label controlled graph grammars, *JCSS* 22 (1981), 144-177.

(6) D. Janssens and G. Rozenberg. A characterization of context-free string languages by directed node-label controlled graph grammars, *Acta Informatica* 16 (1981), 63-85.

(7) D. Janssens and G. Rozenberg. Graph grammars with neighbourhood controlled embedding, *TCS* 21 (1982), 55-74.

(8) D. Janssens and G. Rozenberg. Graph grammars with node-label controlled rewriting and embedding, Report 83-10, Institute of Applied Mathematics and Computer Science, University of Leiden, Wassenaarseweg 80, Leiden, The Netherlands (1983).

(9) M. Nagl. A tutorial and bibliographical survey on graph grammars, in: *Graph-Grammars and Their Application to Computer Science and Biology* (V. Claus, H. Ehrig and G. Rozenberg, eds), LNCS 73, Springer-Verlag, Berlin (1978), 70-126.